CRIAÇÃO DE MODELO DE MACHINE LEARNING

Identificação de spam em emails

Aluno(a): Lucas Henrique Berti Lima
Professor(a): Randerson Oliveira Melville Reboucas
Curso: Ciências da computação
Data: 12, 2024

Sumário

1. Descrição do Problema, Dataset e Pré-Processamento

1.1. Descrição do Problema

O objetivo deste trabalho é criar um modelo de machine learning capaz de classificar emails como spam ou ham(não spam). Emails spam são um problema hoje em dia, pois geralmente eles incluem links perigosos, tentativas de phishing, etc. Isso faz com que seja necessário a utilização de algoritmos que classifiquem automaticamente um email como spam ou não.

1.2. Dataset

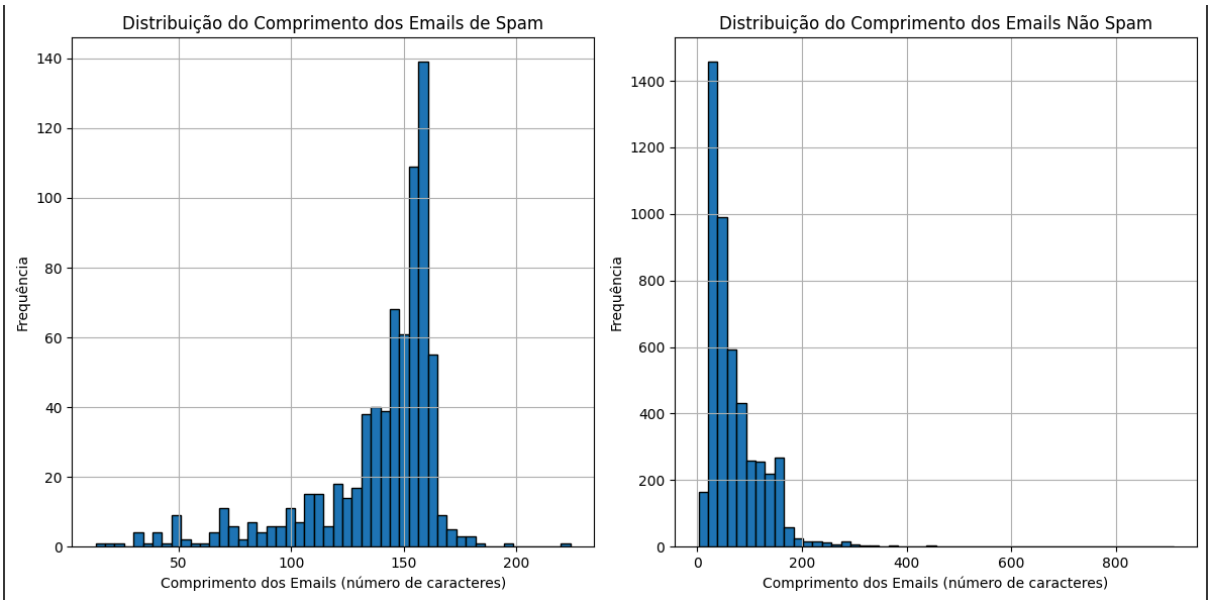
Os dados utilizados para o treino e teste do modelo são originários de dois datasets públicos:

1. **SMS Spam Collection Dataset**
    - **Tamanho:** 5.572 linhas, 4 colunas.
    - **Colunas:**
        - v1 (renomeada para “classificação”): Contém a classificação entre Spam e Ham.
        - v2 (renomeada para “email”): Contém o texto do email.
        - v3 e v4 (excluídas): Predominantemente nulas, sem relevância para o modelo.
2. **Email Classification (Ham-Spam)**
    - **Tamanho:** 179 linhas, 2 colunas.
    - **Colunas:**
        - `email`: Texto do email.
        - `label`: Classificação entre Spam e Ham.

O SMS Spam Collection Dataset tem um tamanho razoável e aparenta ser balanceado, ele continha duas colunas, v3 e v4, que eram na maior parte totalmente nulas, com apenas algumas linhas de texto, por isso optei por excluir elas do

dataset, já que não iriam providenciar nenhuma agregação ao treinamento do modelo.

A seguir fiz algumas análises para entender como os dados estão distribuídos entre a classificação de spam e ham.



Como visto no primeiro gráfico, os emails de spam tendem a serem maiores e os emails que não são spam(ham) são menores.Também foi feito uma análise nas palavras mais frequentes de cada classificação, os emails de spam contém em sua maioria palavras como free, text, call, o que podem indicar que geralmente os emails spam tem um contexto de ganhar algo de graça e enviar um texto ou chamar alguma outra pessoa no email, enquanto nos emails ham as palavras são you, and, it, but, indicando que são mais contextuais para uma conversa entre duas pessoas.

1.3. Pré-Processamento

O texto dos e-mails precisou ser processado para que o modelo conseguisse interpretá-lo de forma eficiente. As seguintes etapas foram realizadas:

1. **Conversão para minúsculas:** Para padronizar o texto e evitar distinções entre palavras maiúsculas e minúsculas, como "FREE" e "free".
2. **Remoção de caracteres especiais:** Símbolos como "!", "@" e "#" foram eliminados, pois não agregam valor relevante para o problema.
3. **Tokenização:** Divisão do texto em palavras individuais para que cada termo pudesse ser analisado separadamente.
4. **Remoção de stopwords:** Palavras comuns como "the", "is" e "and" foram excluídas, pois não contribuem para a identificação de spam.
5. **Lematização:** As palavras foram reduzidas às suas formas básicas, como transformar "running" em "run".
6. **Vetorização com TF-IDF:** O texto foi convertido para uma representação numérica, considerando a frequência das palavras e sua importância relativa.

1.4. Separação de teste e treino

Após todo o tratamento feito na coluna de email, é hora de seguir para a separação entre teste e treino, nesse modelo a separação foi de 80% para treino, que vai ser a parte do dataset que o modelo vai usar para treinar, e 20% para teste, que o modelo vai utilizar para avaliar se está acertando os resultados. Também foi definido o parâmetro random_state=42, que vai garantir que o dataset será dividido da mesma forma toda vez, isso é importante pois se não definirmos isso, a divisão ocorreria de forma aleatória, o que impactaria na capacidade de avaliar os resultados do modelo, pois seriam diferentes cada vez que fosse treinado.

---

2. Resultados do Modelo Base e Aprimorado, com Análise Comparativa e métricas de desempenho

2.1 Métricas de Desempenho

As principais métricas utilizadas foram:

- **Acurácia:** Mede a proporção de previsões corretas (positivas e negativas) em relação ao total de previsões realizadas.
- **Precisão:** Calcula a proporção de previsões positivas corretas entre todas as previsões positivas realizadas pelo modelo. Ajuda a avaliar se o modelo consegue evitar os falsos positivos
- **Recall:** Mede a proporção de previsões positivas corretas entre todas as instâncias que realmente pertencem à classe positiva. Ajuda a avaliar se o modelo consegue evitar os falsos negativos
- **F1-Score:** É a média harmônica entre precisão e recall, oferecendo uma visão equilibrada entre evitar falsos positivos e falsos negativos.

2.2. Modelo Base

O modelo base inicial utilizado foi o KNN, um modelo simples e eficiente para bases de dados mais pequenas. O nosso dataset é de tamanho pequeno para média, possui apenas duas colunas e quase seis mil linhas, então acredito que o KNN seria um bom modelo inicial para testar o agente, visto que ele não requer um poder de processamento muito grande

```
Acurácia: 0.905829596412556
Relatório de classificação:
              precision    recall  f1-score   support

         ham       0.90      1.00      0.95       965
        spam       1.00      0.30      0.46       150

    accuracy                           0.91      1115
   macro avg       0.95      0.65      0.70      1115
weighted avg       0.92      0.91      0.88      1115

Matriz de confusão:
 [[965   0]
 [105  45]]
```

No nosso primeiro modelo, obtivemos um resultado geral médio, com uma acurácia de 90%, uma precisão de 90% e recall de 100% para ham, porém para spam a precisão está em 100% e a recall está em 30%, o que significa que muitos emails do tipo spam não estão sendo classificados como spam, mas quando o modelo diz que um é spam, é 100% de certeza que ele realmente é. Como identificar spam é o objetivo deste trabalho, o resultado desse primeiro modelo é considerado ruim, mesmo que ele consiga identificar email da categoria ham corretamente.

2.3. Testando outros modelos

Como o KNN não conseguiu obter resultados desejados significativos, vamos tentar treinar o modelo com outros modelos de machine learning. Para esta próxima etapa foram selecionados 3 novos modelos: Primeiramente o MultinomialNB, em sequência o SVC e por último o RandomForest.

Resultado do teste com o modelo MultinomialNB

```
Acurácia: 0.9614349775784753
Relatório de classificação:
              precision    recall  f1-score   support

         ham       0.96      1.00      0.98       965
        spam       1.00      0.71      0.83       150

    accuracy                           0.96      1115
   macro avg       0.98      0.86      0.91      1115
weighted avg       0.96      0.96      0.96      1115

[[965   0]
 [ 43 107]]
```

Com o MultinomialNB, podemos ver de cara que o resultado foi muito melhor que o KNN, principalmente na questão de identificar spam, visto que o recall aumentou de 30% para 71%.

Resultado do teste com o modelo SVC

```
Acurácia: 0.9730941704035875
Relatório de classificação:
              precision    recall  f1-score   support

         ham       0.97      1.00      0.98       965
        spam       0.98      0.82      0.89       150

    accuracy                           0.97      1115
   macro avg       0.97      0.91      0.94      1115
weighted avg       0.97      0.97      0.97      1115

[[962   3]
 [ 27 123]]
```

Com o SVC, obtivemos um resultado ainda melhor, conseguindo 10% a mais no recall

Resultado do teste com o modelo Random Forest

```
Acurácia: 0.9730941704035875
Relatório de classificação:
              precision    recall  f1-score   support

         ham       0.97      1.00      0.98       965
        spam       1.00      0.80      0.89       150

    accuracy                           0.97      1115
   macro avg       0.98      0.90      0.94      1115
weighted avg       0.97      0.97      0.97      1115

[[965   0]
 [ 30 120]]
```

O Random Forest obteve resultados menores que o SVC, mas ainda é o segundo melhor modelo comparado aos outros

Como podemos ver, o modelo com o melhor resultado foi o SVC, eu esperava que o modelo do RandomForest fosse ter o melhor resultado, pois é o que tem os parâmetros mais altos, mas ele acabou perdendo por pouco. Pesquisando sobre isso, descobri que o SVC é mais eficiente para dados classificatórios, fazendo com que ele consiga definir uma margem de separação entre as classes melhor.

2.4. Aprimorando melhores modelos

Como foi visto na comparação, os melhores modelos foram o SVC(Support Vector Machines) e o Random Forest, então vamos selecionar esses dois modelos e tentar melhorar os resultados deles através da hiper parametrização via grid search, o grid search vai basicamente testar as diferentes combinações de hiperparâmetros passados no código, a fim de encontrar a melhor combinação para o modelo.

Para o RandomForest foi definido os seguintes parâmetros: n_estimators = 35, 55, 75 que seria o numero de arvores; max_depth = 30, 60, 75 que representa a profundidade de cada árvore; min_samples_split = 5, 10 ,15 que representa o número mínimo de amostras necessárias para dividir um nó
O resultado foi:

```
Melhores hiperparâmetros: {'max_depth': 75, 'min_samples_split': 15, 'n_estimators': 75}
Acurácia com modelo otimizado: 0.9775784753363229
Relatório de classificação:
              precision    recall  f1-score   support

         ham       0.97      1.00      0.99       965
        spam       1.00      0.83      0.91       150

    accuracy                           0.98      1115
   macro avg       0.99      0.92      0.95      1115
weighted avg       0.98      0.98      0.98      1115

[[965   0]
 [ 25 125]]
```

como podemos ver, o Random forest obteve um aumento de 3% no recall, o que ajudou na detecção correta de emails spam

Para o SVC foi definido os seguintes parâmetros: C = 0.1, 1, 10, 100 que seria o parâmetro de regularização; gamma = 0.01, 0.1, 1, 10 que seria um parâmetro para modelos não lineares; kernel = sigmoid, linear, poly, rbf o kernel seria basicamente a função para realizar a separação dos dados.
O resultado foi:

```
Melhores hiperparâmetros: {'C': 100, 'gamma': 0.1, 'kernel': 'sigmoid'}
Acurácia: 0.9766816143497757
Relatório de classificação:
              precision    recall  f1-score   support

         ham       0.98      1.00      0.99       965
        spam       0.98      0.85      0.91       150

    accuracy                           0.98      1115
   macro avg       0.98      0.92      0.95      1115
weighted avg       0.98      0.98      0.98      1115

Matriz de confusão:
 [[962   3]
 [ 23 127]]
```

2.5. Testando os modelos aprimorados com um outro dataset

A seguir eu utilizei um outro dataset para testar o quão bem os dois modelos se sairiam com um conjunto de dados diferente do utilizado no treino e teste. Como é uma base de dados diferente, o jeito que os emails estão escritos pode ser um pouco diferente, o que pode causar dificuldades para o modelo, mesmo que ele tenha um bom desempenho na base de dados atual.

Resultados do teste com o melhor modelo do Random Forest:

```
Resultados do modelo RandomForestClassifier(max_depth=75, min_samples_split=15, n_estimators=75,
                       random_state=42)
Acurácia para ham: 98.00%
Acurácia para spam: 36.71%
```

Como podemos ver, obtivemos uma acurácia muito alta para identificar e mails ham, mas para e mails spam a acurácia ficou baixa

Resultados do teste com o melhor modelo do SVC:

```
Resultados do modelo SVC(C=100, gamma=0.1, kernel='sigmoid')
Acurácia para ham: 86.00%
Acurácia para spam: 53.16%
```

Já no modelo SVC, a acurácia para ham ficou mais baixa, porém a acurácia para identificar emails spam ficou em mais de 50%

2.6. Implementação

Adicionei também no notebook uma célula de código que permite que seja digitado um email manualmente, assim permitindo que o usuário consiga testar qualquer email que ele deseje. Basicamente o código vai pegar o input do usuário e realizar o mesmo tratamento que foi feito para o dataset, vetorização, lemmatizer, etc. então ele pega o modelo que foi passado como parâmetro e realiza uma predição no email.

```python
def pre_processamento(email):
  email = email.lower()
  email_sem_especiais = re.sub(r'[^\w\s]', '', email)

  email_tokenizado = word_tokenize(email_sem_especiais)

  email_tokenizado = [word for word in email_tokenizado if word not in stop_words]

  email_tokenizado = [lemmatizer.lemmatize(word) for word in email_tokenizado]

  email_tokenizado = ' '.join(email_tokenizado)

  email_vetorizado = vectorizer.transform([email_tokenizado])

  return email_vetorizado

def testar_modelo(email, melhor_modelo):
  predicao = melhor_modelo.predict(email)
  return predicao


email = input("Cole aqui o email a ser classificado: ")
email_pre_processado = pre_processamento(email)
predicao = testar_modelo(email_pre_processado, best_svc)
print(predicao)
```

---

3. Limitações do Agente e Possíveis Melhorias Futuras

3.1. Limitações

1. **Generalização:** O modelo pode não funcionar bem com e-mails muito diferentes dos dados de treinamento, pois existem milhões de emails spam sendo lançados, então o dataset só vai conter uma fração baixa deles, então o modelo só conseguira identificar emails de spam que sigam o padrão que já tem no dataset.

3.2. Melhorias Futuras

1. **Usar modelos baseados em redes neurais:** Explorar redes como o **BERT**, que entendem melhor o contexto das palavras no texto.

2. **Ampliar o dataset:** Adicionar mais exemplos no dataset, principalmente de emails spam pois assim ele poderia treinar com mais casos diferentes e melhorar a acurácia geral do modelo.
3. **Atualização contínua do modelo:** Novos emails de spam são enviados todos os dias, então uma fonte contínua de novos exemplos para o modelo seria uma ótima melhoria, pois ele poderia treinar com os estilos de spam mais atuais e não cair em defasagem.

---

4. Conclusão

O trabalho desenvolvido neste projeto foi bem interessante para o aprendizado de IA, apesar de ser um projeto mais simples, ele até requer um grau de esforço alto. O modelo escolhido inicialmente já mostrava as maiores dificuldades que seriam enfrentadas, que no caso é a dificuldade de encontrar corretamente os emails spam, porém nos modelos subsequentes, conseguimos melhorar esse resultado, diminuindo bastante a quantidade de emails spam classificados incorretamente. Fizemos ainda um teste com outro conjunto de dados de email, e nele obtivemos uma porcentagem de 50% de acerto para emails spam, no meu entendimento isso se deve ao fato de que o estilo de email pode ser diferente do estilo usado no treinamento do modelo. Por causa disso, o ideal seria se tivesse uma base dados maior, com uma quantidade de spam e ham parecida, e mesmo assim, com o passar do tempo o modelo iria se defasar pois os emails spam vão mudar o jeito que são escritos, o que vai fazer com que o modelo tenha mais dificuldade em identificá los. Mas para um modelo inicial, eu acredito que ele tenha se saído bem.